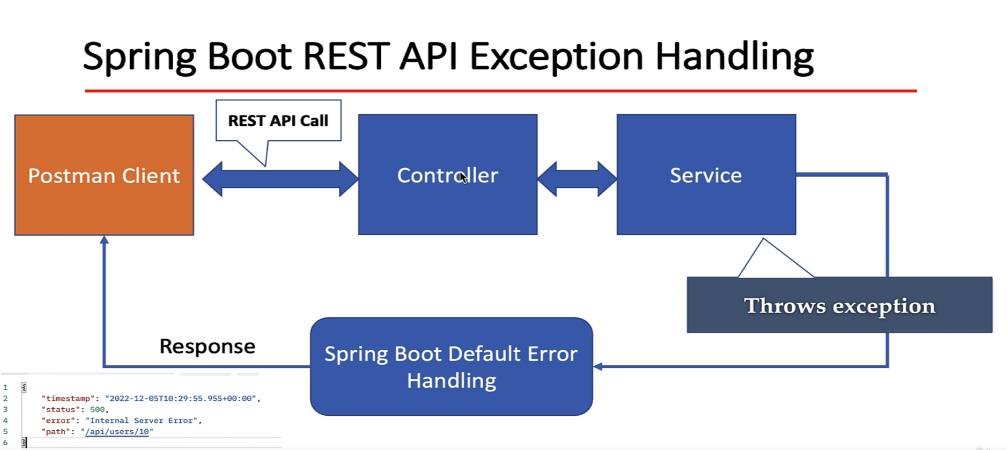
**Spring Default Error Handling**

* 
* Spring are un Default Error Handler, si orice request gresit va arata asa:

{

    "timestamp": "2023-07-30T20:05:04.965+00:00",

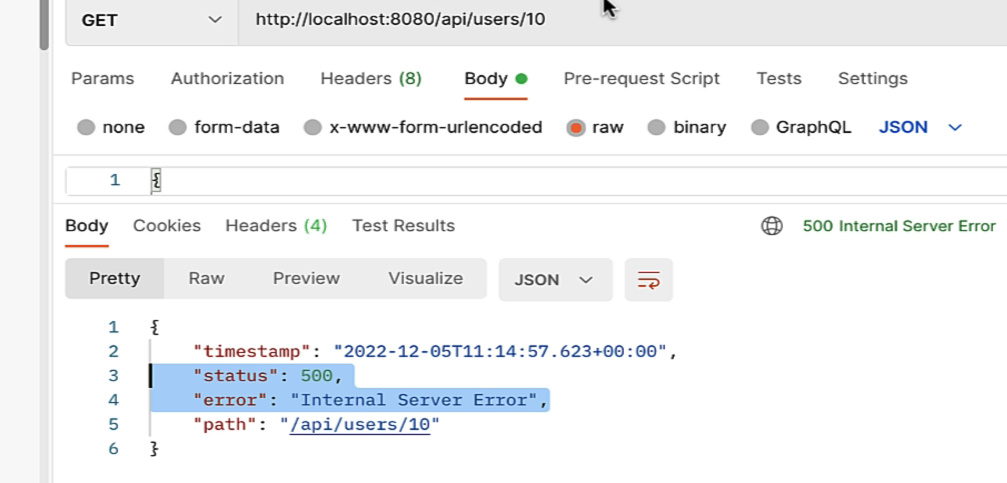
    "status": 404,

    "error": "Not Found",

    "path": "/test"

}

* Dar, daca de ex facem un get cu un id inexistent la un user, vom primi un asa payload:



Dar el nu e prea descriptiv, si nici nu spune vreun mesaj. In plus, nu 500 e error, caci e client error, nu server error.

**ControllerAdvice WebRequest**

* Cand cream un ControllerAdvice, la @ExceptionHandler method dam un parametru de tipul la exceptia primita.
* Totusi, mai putem trimite inca un parametru pentru a primi mai multe informatii despre link trimis de user, si anume WebRequest
* @ControllerAdvice  
  public class GlobalControllerAdvice {  
   @ExceptionHandler(NotFoundResourceException.class)  
   public ResponseEntity<ErrorDetails> notFound(NotFoundResourceException exception, WebRequest webRequest){  
   return new ResponseEntity<>(  
   new ErrorDetails(System.*currentTimeMillis*(),webRequest.getDescription(false),exception.getMessage(),404),  
   HttpStatus.*NOT\_FOUND* );  
   }  
  }

getDescription(boolean) – returneaza path trimis de user

- false – trimitem pur doar url

{

    "timestamp": 1690754569533,

    "path": "uri=/1",

    "message": "User with 1 not found.",

    "error": 404

}

- true – trimitem mai multe detalii despre client, adica asa

{

    "timestamp": 1690754491712,

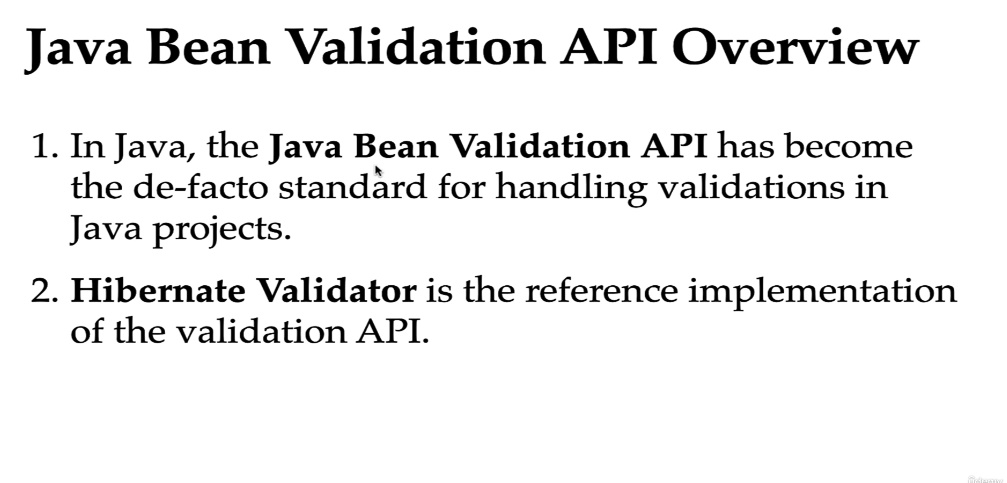
    "path": "uri=/1;client=0:0:0:0:0:0:0:1",

    "message": "User with 1 not found.",

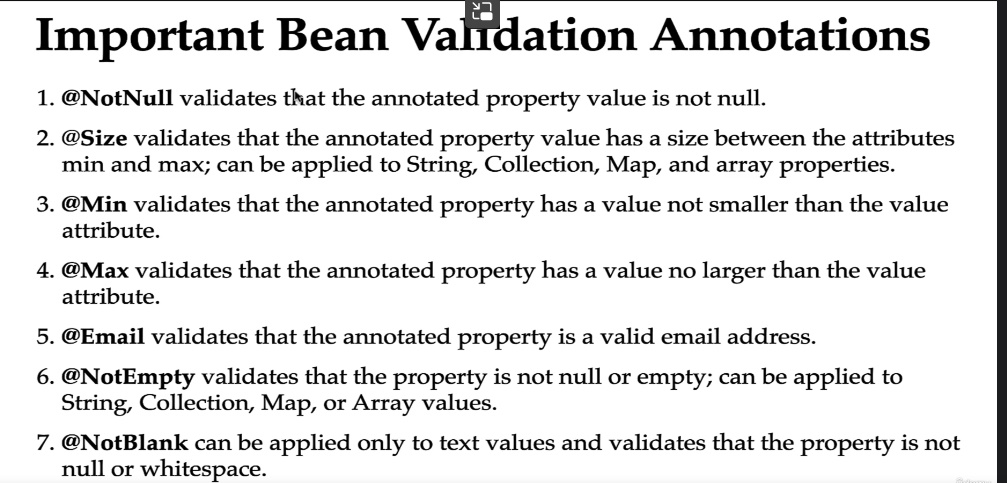
    "error": 404

}

**JavaBeanValidation**



Java Bean Validation API e doar un standart,ca JPA, dar Hibernate Validator este o implementare concreta



@Size poate fi folosit si la colectii

@NotEmpty – valideaza daca nu e null sau goala. Poate fi aplicata si la collectii

@NotBlank – valideaza doar String si se uita sa nu fie null,empty sau doar din whitespace

* Validarea se face mereu la DTO
* Daca nu oferim un obiect de tip BindingResult dupa @Valid, atunci se va arunca o exceptie de tip MethodArgumentNotValidException
* Erorile de validare cel mai bine e sa la punem in Map

**Adaugarea erorilor in Map**

* Putem face direct o metoda cu @ExceptionHandler ce ia o exceptie de tip MethodArgumentNotValidException
* Ea ne ofera si posibilitatea de a lua BindingResult object trimis in exceptie
* BindingResult are metoda **getAllErrors()**, ce returneaza o lista de **ObjectError**, dar defapt ea returneaza **FieldError**, doar ca le face cast in ObjectError. FieldError e o class ce extinde clasa ObjectError. Sunt ambele clase simple.
* Cu ObjectError putem primi mesajul la eroare, nu si numele la field, de aceea facem conversie in FieldError, sa avem acces la **getField()**

@ExceptionHandler(MethodArgumentNotValidException.class)  
public ResponseEntity<Map<String,String>> validationError(MethodArgumentNotValidException exception){  
 Map<String,String> errorMap = new HashMap<>();  
  
 List<ObjectError> errors = exception.getBindingResult().getAllErrors();  
  
 errors.forEach(error -> {  
 String field = ((FieldError) error).getField();  
 String message = error.getDefaultMessage();  
  
 errorMap.put(field,message);  
 });  
  
 return new ResponseEntity<>(errorMap,HttpStatus.*BAD\_REQUEST*);  
}

**ResponseEntityExceptionHandler**

* Clasa data ofera multe metode pentru a returna niste raspunsuri pentru diferite tipuri de exceptii posibile la requesturi in RestController
* @ControllerAdvice  
  public class GlobalControllerAdvice extends ResponseEntityExceptionHandler {  
   @Override  
   public ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException exception, HttpHeaders headers, HttpStatusCode status, WebRequest request) {  
   Map<String,String> errorMap = new HashMap<>();  
    
   List<ObjectError> errors = exception.getBindingResult().getAllErrors();  
    
   errors.forEach(error -> {  
   String field = ((FieldError) error).getField();  
   String message = error.getDefaultMessage();  
    
   errorMap.put(field,message);  
   });  
    
   return new ResponseEntity<>(errorMap,HttpStatus.*BAD\_REQUEST*);  
   }  
    
  }

Nu mai punem @ExceptionHandler!!!!